

5 interleaved, as shown in Figure 3c by matrices 30a and 30b, having rows 34a and 34b, respectively. Accordingly, datapaths made of cells of different widths can be efficiently placed. Using this method, complex datapaths can be built using basic standard cells, without the need for custom cell development. It should be noted that two or more matrices may be interleaved using interleaved columns,  
10 as well as interleaved rows as shown in Figure 3c.

When timing-driven tools have difficulties with complex constraints, the datapath matrix (or matrices) can ensure timings for the critical paths. By adjusting column and row spacing, free space 38 can be planned within the matrix to allow timing-driven placement of embedded standard cells along with  
15 the structured placement cells. Figure 4 illustrates free space 38 within one or more matrices for further placement of timing-driven unstructured cells. It has been found that this approach provides improved density by increasing the percentage of area utilization (by gates) compared to an approach where datapath blocks are placed as embedded hard macros.

20 Implementation of the method of Figure 2 requires three parts: logic synthesis, datapath description file, and placement automation. These aspects are described below.

To have more control over the way the logic synthesis is done to achieve the expected netlist, datapaths are described as dedicated modules. Also, each  
25 datapath basic element is described as a "box", either containing a basic RTL (register transfer level) description for the involved function, or direct instantiation of the involved cell.

All datapath modules are instantiated within the RTL description of the integrated circuit, leading to a structural type of description. Once the RTL  
30 description is ready, logic synthesis is performed using a bottom-up approach:

5 datapath basic elements are synthesized first and the top level is synthesized using the "fixed" attribute on basic element instances. For accurate timing analysis, the net loads corresponding to structural placements are annotated either from the previous layout run or from estimated load values.

10 Once the DSP core netlist is ready, the involved instances can be identified and collected through wildcards and a description file for the corresponding matrix can be easily built. The syntax of the description file is very simple and the requested information about the involved matrices are: matrix name, number of rows, number of columns, space between rows, space between columns, matrix location, strap pitch and involved instances per row. Strap  
15 pitch allows planning for vertical power ground straps by adding convenient space at locations within the matrices.

Location can be absolute or relative to the location of another matrix. Relative location is useful for cases of further floorplan updates. If the floorplan is changed and if the location of all matrices depends on the location of the  
20 reference matrix, then the only requested manual change will be to move the latter matrix. Also, this feature allows the designer to try various datapath implementations, since the cost of manual intervention is very low.

Rows are concisely described using patterns and indices. Three examples are:

25 ROWx="pattern" index\_start= index\_end= index step= pattern\_width=  
ROWx="pattern1 pattern2" index\_start= index\_end= index step=  
pattern\_width=  
ROWx="pattern1" index\_start= index\_end= index step= pattern\_width=,  
"pattern2" index\_start= index\_end= index step= pattern\_width=,  
30 "pattern3" index\_start= index\_end= index step= pattern\_width=

- 5 From this description file, the Datapath generator produces a full scheme language configuration file for involved matrices, allowing installation of structured placement within the floorplan.

An example of a description file is:

```

    ROW0='acc0_reg_@INDEX@/data_reg_reg" INDEX_start=0 INDEX_end=38
10 INDEX_step=2 PATTERN_WIDTH=66.3
    ROW1='acc1_reg_@INDEX@/data_reg_reg" INDEX_start=0 INDEX_end=38
    INDEX_step=2 PATTERN_WIDTH=66.3
    ROW2='acc2_reg_@INDEX@/data_reg_reg" INDEX_start=0 INDEX_end=38
    INDEX_step=2 PATTERN_WIDTH=66.3
15    ROW3='acc3_reg_@INDEX@/data_reg_reg" INDEX_start=0 INDEX_end=38
    INDEX_step=2 PATTERN_WIDTH=66.3
    ROW4='acc0_reg_@INDEX@/data_reg_reg" INDEX_start=39 INDEX_end=1
    INDEX_step=-2 PATTERN_WIDTH=66.3
    ROW1='acc1_reg_@INDEX@/data_reg_reg" INDEX_start=39 INDEX_end=1
20 INDEX_step=-2 PATTERN_WIDTH=66.3
    ROW2='acc2_reg_@INDEX@/data_reg_reg" INDEX_start=39 INDEX_end=1
    INDEX_step=-2 PATTERN_WIDTH=66.3
    ROW3='acc3_reg_@INDEX@/data_reg_reg" INDEX_start=39 INDEX_end=1
    INDEX_step=-2 PATTERN_WIDTH=66.3

```

- 25 The eight-line description set forth above provides placement for 152 cells within the regular structure.

Once all description files are generated for the datapaths, the datapath generator 14 passes the language configuration file to the place and route tool 12 to install all datapath logic within the floorplan, taking into account